Chains of Reasoning over Entities, Relations, and Text using Recurrent Neural Networks

Rajarshi Das, Arvind Neelakantan, David Belanger, Andrew McCallum
College of Information and Computer Sciences
University of Massachusetts, Amherst
{rajarshi, arvind, belanger, mccallum}@cs.umass.edu

Abstract

Our goal is to combine the rich multi-step inference of symbolic logical reasoning with the generalization capabilities of neural networks. We are particularly interested in complex reasoning about entities and relations in text and large-scale knowledge bases (KBs). Neelakantan et al. (2015) use RNNs to obtain dense representations of multi-hop paths in KBs; however for multiple reasons, the approach lacks accuracy and practicality. This paper proposes three significant modeling advances: (1) we learn to jointly reason about relations, *entities, and entity-types*; (2) we use neural attention modeling to incorporate *multiple paths*; (3) we learn to *share strength in a single RNN* that represents logical composition across all relations. On a large-scale Freebase+ClueWeb prediction task, we achieve 25% error reduction, and a 53% error reduction on sparse relations. On chains of reasoning in WordNet we reduce error in mean quantile by 84% versus the previous state of the art.[1].

1 Introduction

There is a rising interest in extending neural networks to perform more complex reasoning, formerly addressed only by symbolic and logical reasoning systems. So far this work has mostly focused on small or synthetic data (Grefenstette, 2013; Bowman et al., 2014; Rocktäschel and Riedel, 2016). Our interest is primarily in reasoning about large knowledge bases (KBs) with diverse semantics, populated from text. One method for populating a KB from text (and for representing diverse semantics in the KB) is *Universal Schema* (Riedel et al., 2013; Verga et al., 2016), which learns vector embeddings of relation types - the union of all input relation types, both from the schemas of multiple structured KBs, as well as expressions of relations in natural language text.

i. place.birth$(a, b) \leftarrow$ 'was_born_in'$(a, x) \wedge$ '$commonly\_known\_as$'(x, b)
ii. location.contains$(a, b) \leftarrow$ nationality$^{-1}(a, x)$ $\wedge$ place.birth(x, b)
iii. book.characters$(a, b) \leftarrow$'aka'$(a, x) \wedge$ (theater.character.plays)$^{-1}(x, b)$
iv. cause.death$(a, b) \leftarrow$'$contracted$'(a, b)

Table 1: Several highly probable clauses learnt by our model. The textual relations are shown in quotes and italicized. Our model has the ability to combine textual and schema relations. r^{-1} is the inverse relation r, i.e. $r(a, b) \Leftrightarrow r^{-1}(b, a)$.

An important reason to populate a KB is to support not only look-up-style question answering, but reasoning on its entities and relations in order to make inferences not directly stored in the KB. KBs are often highly incomplete (Min et al., 2013), and reasoning can fill in these missing facts. The "matrix completion" mechanism that underlies the common implementation of Universal Schema can thus be seen as a simple type of reasoning, as can other work in tensor factorization (Nickel et al., 2011; Bordes et al., 2013; Socher et al., 2013). However these methods can be understood as operating on single pieces of evidence: for example, inferring that Microsoft–*located-in*–Seattle implies Microsoft–*HQ-in*–Seattle.

A highly desirable, richer style of reasoning

[1]The code and data are available at https://rajarshd.github.io/ChainsofReasoning/

Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers, pages 132–141,
Valencia, Spain, April 3-7, 2017.

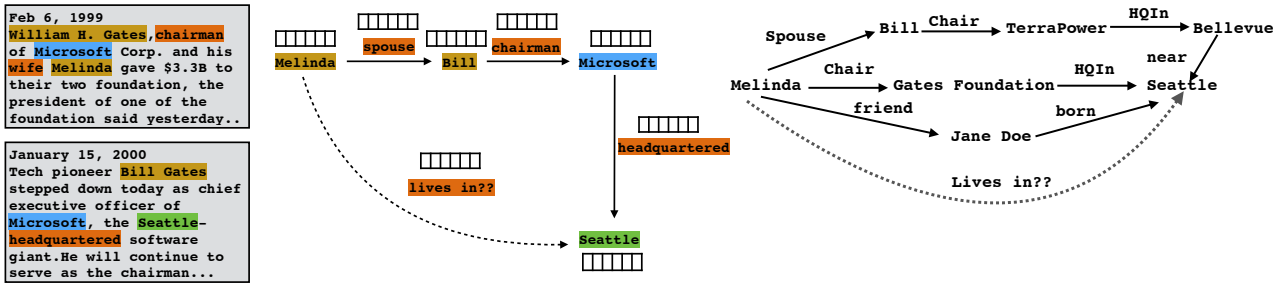

Figure 1: The nodes in the knowledge graphs represent entities and the labeled edges represent relations. (a) A path between 'Melinda' and 'Seattle' combining relations from two different documents. (b) There are multiple paths between entities in a knowledge graph. The top two paths are predictive of the fact that Melinda may '*live in*' Seattle, but the bottom (fictitious) path isn't.

makes inferences from Horn clauses that form multi-hop paths containing three or more entities in the KB's entity-relation graph. For example, we may have no evidence directly linking Melinda Gates and Seattle. However, we may infer with some likelihood that Melinda–*lives-in*–Seattle, by observing that the KB contains the path Melinda–*spouse*–Bill–*chairman*–Microsoft–*HQ-in*–Seattle (Fig. 1a).

Symbolic rules of this form are learned by the Path Ranking Algorithm (PRA) (Lao et al., 2011). Dramatic improvement in generalization can be obtained by reasoning about paths, not in terms of relation-symbols, but Universal Schema style relation-vector-embeddings. This is done by Neelakantan et al. (2015), where RNNs compose the per-edge relation embeddings along an arbitrary-length path, and output a vector embedding representing the inferred relation between the two entities at the end-points of the path. This approach thus represents a key example of complex reasoning over Horn clause chains using neural networks. However, for multiple reasons detailed below it is inaccurate and impractical.

This paper presents multiple modeling advances that significantly increase the accuracy and practicality of RNN-based reasoning on Horn clause chains in large-scale KBs. (1) Previous work, including (Lao et al., 2011; Neelakantan et al., 2015; Guu et al., 2015) reason about chains of relations, but not the entities that form the nodes of the path. In our work, we jointly learn and reason about relation-types, entities, and entity-types. (2) The same previous work takes only a single path as evidence in inferring new predictions. However, as shown in Figure 1b, multiple paths can provide evidence for a prediction. In our work, we use neural attention mechanisms to reason about multiple paths. We use a pooling function which does soft attention during *gradient step* and find it to work better. (3) The most problematic impracticality of the above previous work[2] for application to KBs with broad semantics is their requirement to train a separate model for each relation-type to be predicted. In contrast, we train a single, high-capacity RNN that can predict all relation types. In addition to efficiency advantages, our approach significantly increases accuracy because the multi-task nature of the training shares strength in the common RNN parameters.

We evaluate our new approach on a large scale dataset of Freebase entities, relations and ClueWeb text. In comparison with the previous best on this data, we achieve an error reduction of 25% in mean average precision (MAP). In an experiment specially designed to explore the benefits of sharing strength with a single RNN, we show a 54% error reduction in relations that are available only sparsely at training time. We also evaluate on a second data set, chains of reasoning in WordNet. In comparison with previous state-of-the-art (Guu et al., 2015) our model achieves a 84% reduction in error in mean quantile.

2 Background

In this section, we introduce the compositional model (Path-RNN) of Neelakantan et al. (2015). The Path-RNN model takes as input a path between two entities and infers new relations between them. Reasoning is performed non-

[2] with exception of (Guu et al., 2015)

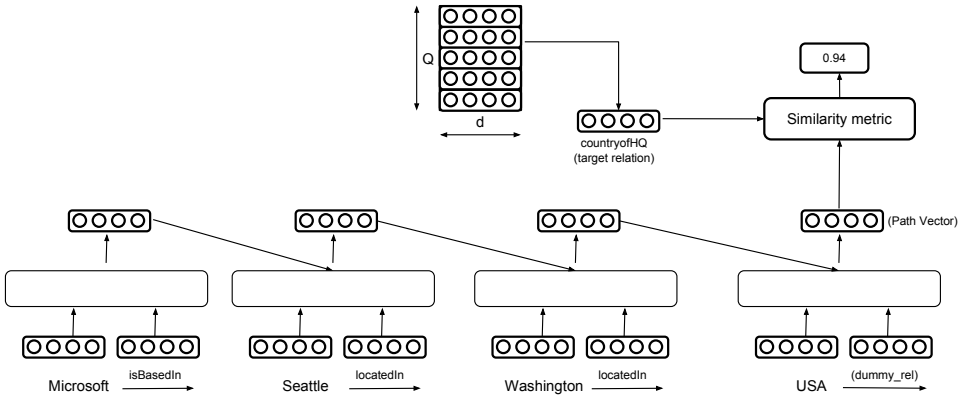

Figure 2: At each step, the RNN consumes both entity and relation vectors of the path. The entity representation can be obtained from its types. The path vector $\mathbf{y}_\pi$ is the last hidden state. The parameters of the RNN and relation embeddings are shared across all query relations. The dot product between the final representation of the path and the query relation gives a confidence score, with higher scores indicating that the query relation exists between the entity pair.

atomically about conjunctions of relations in an arbitrary length path by composing them with a recurrent neural network (RNN). The representation of the path is given by the last hidden state of the RNN obtained after processing all the relations in the path.

Let (e_s, e_t) be an entity pair and $\mathcal{S}$ denote the set of paths between them. The set $\mathcal{S}$ is obtained by doing random walks in the knowledge graph starting from e_s till e_t. Let $\pi = \{e_s, r_1, e_1, r_2, \ldots, r_k, e_t\} \in \mathcal{S}$ denote a path between (e_s, e_t). The length of a path is the number of relations in it, hence, $(\text{len}(\pi) = k)$. Let $\mathbf{y_{r_t}} \in \mathbb{R}^d$ denote the vector representation of r_t. The Path-RNN model combines all the *relations* in π sequentially using a RNN with an intermediate representation $\mathbf{h_t} \in \mathbb{R}^{\mathbf{h}}$ at step t given by

$$\mathbf{h_t} = f(\mathbf{W^r_{hh}} \mathbf{h_{t-1}} + \mathbf{W^r_{ih}} \mathbf{y^r_{r_t}}). \quad (1)$$

$\mathbf{W^r_{hh}} \in \mathbb{R}^{h \times h}$ and $\mathbf{W^r_{ih}} \in \mathbb{R}^{d \times h}$ are the parameters of the RNN. Here r denotes the query relation. Path-RNN has a specialized model for predicting each query relation r, with separate parameters $(\mathbf{y^r_{r_t}}, \mathbf{W^r_{hh}}, \mathbf{W^r_{ih}})$ for each r. f is the sigmoid function. The vector representation of path π $(\mathbf{y}_\pi)$ is the last hidden state $\mathbf{h_k}$. The similarity of $\mathbf{y}_\pi$ with the query relation vector $\mathbf{y_r}$ is computed as the dot product between them:

$$\text{score}(\pi, \text{r}) = \mathbf{y}_\pi \cdot \mathbf{y_r} \quad (2)$$

Pairs of entities may have several paths connecting them in the knowledge graph (Figure 1b). Let $\{s_1, s_2, \ldots, s_N\}$ be the similarity scores (Equation 2) for N paths connecting an entity pair (e_s, e_t). Path-RNN computes the probability that the entity pair (e_s, e_t) participates in the query relation (r) by,

$$\mathbb{P}(r|e_s, e_t) = \quad \sigma(\max(s_1, s_2, \ldots, s_N)) \quad (3)$$

where σ is the *sigmoid* function.

Path-RNN and other models such as the Path Ranking Algorithm (PRA) and its extensions (Lao et al., 2011; Lao et al., 2012; Gardner et al., 2013; Gardner et al., 2014) makes it impractical to be used in downstream applications, since it requires training and maintaining a model for each relation type. Moreover, parameters are not shared across multiple target relation types leading to large number of parameters to be learned from the training data.

In (3), the Path-RNN model selects the maximum scoring path between an entity pair to make a prediction, possibly ignoring evidence from other important paths. Not only is this a waste of computation (since we have to compute a forward pass for all the paths anyway), but also the relations in all other paths do not get any gradients updates during training as the max operation returns zero gradient for all other paths except the maximum scoring one. This is especially ineffective during the initial stages of the training since the maximum probable path will be random.

The Path-RNN model and other multi-hop relation extraction approaches (such as Guu et al. (2015)) ignore the entities in the path. Consider the following paths JFK–*locatedIn*–

NYC–*locatedIn*–NY and Yankee Stadium–*locatedIn*–NYC–*locatedIn*–NY. To predict the $airport\_serves$ relation, the Path-RNN model assigns the same scores to both the paths even though the first path should be ranked higher. This is because the model does not have information about the entities and just uses the relations in the path for prediction.

3 Modeling Approach

3.1 Shared Parameter Architecture

Previous section discussed the problems associated with per-relation modeling approaches. In response, we *share* the relation type representation and the composition matrices of the RNN across all target relations enabling lesser number of parameters for the same training data. We refer to this model as *Single-Model*. Note that this is just *multi-task learning* (Caruana, 1997) among prediction of target relation types with an underlying shared parameter architecture. The RNN hidden state in (1) is now given by:

$$\mathbf{h_t} = f(\mathbf{W_{hh}}\mathbf{h_{t-1}} + \mathbf{W_{ih}}\mathbf{y_{r_t}}). \quad (4)$$

Readers should take note that the parameters here are independent of each target relation r.

Model Training

We train the model using existing observed facts (triples) in the KB as positive examples and unobserved facts as negative examples. Let $\mathcal{R} = \{\gamma_1, \gamma_2, \ldots, \gamma_n\}$ denote the set of all query relation types that we train for. Let $\Delta^+_{\mathcal{R}}, \Delta^-_{\mathcal{R}}$ denote the set of positive and negative triples for *all* the relation types in $\mathcal{R}$. The parameters of the model are trained to minimize the negative log-likelihood of the data.

$$L(\Theta, \Delta^+_{\mathcal{R}}, \Delta^-_{\mathcal{R}}) = -\frac{1}{M} \sum_{e_s, e_t, r \in \Delta^+_{\mathcal{R}}} \log \mathbb{P}(r|e_s, e_t) + \sum_{\hat{e}_s, \hat{e}_t, \hat{r} \in \Delta^-_{\mathcal{R}}} \log(\, 1 - \mathbb{P}(\hat{r}|\hat{e}_s, \hat{e}_t)) \quad (5)$$

Here M is the total number of training examples and Θ denotes the set of all parameters of the model (lookup table of embeddings (shared) and parameters of the RNN (shared)). It should be noted that the Path-RNN model has a separate loss function for each relation $r \in \mathcal{R}$ which depends only on the relevant subset of the data.

3.2 Score Pooling

In this section, we introduce new methods of score pooling that takes into account multiple paths between an entity pair. Let $\{s_1, s_2, \ldots, s_N\}$ be the similarity scores (Equation 2) for N paths connecting an entity pair (e_s, e_t). The probability for entity pair (e_s, e_t) to participate in relation r (Equation 3) is now given by,

1. Top-(k): A straightforward extension of the 'max' approach in which we average the top k scoring paths. Let $\mathcal{K}$ denote the indices of top-k scoring paths.

$$\mathbb{P}(r|e_s, e_t) = \sigma(\frac{1}{k}\sum_j s_j), \forall j \in \mathcal{K}$$

2. Average: Here, the final score is the average of scores of all the paths.

$$\mathbb{P}(r|e_s, e_t) = \sigma(\frac{1}{N}\sum_{i=1}^{N} s_i)$$

3. LogSumExp: In this approach the pooling layer is a smooth approximation to the 'max' function - LogSumExp (LSE). Given a vector of scores, the LSE is calculated as

$$\mathrm{LSE}(s_1, s_2, \ldots, s_N) = \log(\sum_i \exp(s_i))$$

and hence the probability of the triple is,

$$\mathbb{P}(r|e_1, e_2) = \sigma(\mathrm{LSE}(s_1, s_2, \ldots, s_N))$$

The average and the LSE pooling functions apply non-zero weights to *all* the paths during inference. However only a few paths between an entity pair are predictive of a query relation. LSE has another desirable property since $\frac{\partial \mathrm{LSE}}{\partial s_i} = \frac{\exp(s_i)}{\sum_i \exp(s_i)}$. This means that during the back-propagation step, every path will receive a share of the gradient proportional to its score and hence this is a kind of attention during the gradient step. In contrast, for averaging, every path will receive equal $(\frac{1}{N})$ share of the gradient. Top-(k) (similar to max) receives sparse gradients.

3.3 Incorporating Entities

A straightforward way of incorporating entities is to include entity representations (along with relations) as input to the RNN. Learning separate

representations of entity, however has some disadvantages. The distribution of entity occurrence is heavy tailed and hence it is hard to learn good representations of rarely occurring entities. To alleviate this problem, we use the entity types present in the KB as described below.

Most KBs have annotated types for entities and each entity can have multiple types. For example, Melinda Gates has types such as *CEO, Duke University Alumni, Philanthropist, American Citizen* etc. We obtain the entity representation by a simple addition of the entity type representations. The entity type representations are learned during training. We limit the number of entity types for an entity to 7 most frequently occurring types in the KB. Let $\mathbf{y_{e_t}} \in \mathbb{R}^m$ denote the representation of entity e_t, then 4 now becomes

$$\mathbf{h_t} = f(\mathbf{W_{hh}h_{t-1}} + \mathbf{W_{ih}y_{r_t}} + \mathbf{W_{eh}y_{e_t}}) \quad (6)$$

$\mathbf{W_{eh}} \in \mathbb{R}^{m \times h}$ is the new parameter matrix for projecting the entity representation. Figure 2 shows our model with an example path between entities (*Microsoft, USA*) with *countryOfHQ* (country of head-quarters) as the query relation.

4 Related Work

Two early works on extracting clauses and reasoning over paths are SHERLOCK (Schoenmackers et al., 2010) and the Path Ranking Algorithm (PRA) (Lao et al., 2011). SHERLOCK extracts purely symbolic clauses by exhaustively exploring relational paths of increasing length. PRA replaces exhaustive search by random walks. Observed paths are used as features for a per-target-relation binary classifier. Lao et al. (2012) extend PRA by augmenting KB-schema relations with observed text patterns. However, these methods do not generalize well to millions of distinct paths obtained from random exploration of the KB, since each unique path is treated as a singleton, where no commonalities between paths are modeled. In response, pre-trained vector representations have been used in PRA to tackle the feature explosion (Gardner et al., 2013; Gardner et al., 2014) but still rely on a classifier using atomic path features. Yang et al. (2015) also extract horn rules, but they restrict it to a length of 3 and the literals are restricted to schema types in the knowledge base. Zeng et al. (2016) show improvements in relation extraction by incorporating sentences which contain one entity.

Stats	#
# Freebase relation types	27,791
# textual relation types	23,599
# query relation types	46
# entity pairs	3.22M
# unique entity types	2218
Avg. path length	4.7
Max path length	7
Total # paths	191M

Table 2: Statistics of the dataset.

Guu et al. (2015) introduce new compositional techniques by modeling additive and multiplicative interactions between relation matrices in the path. However they model only a *single* path between an entity pair in-contrast to our ability to consider multiple paths. Toutanova et al. (2016) improves upon them by additionally modeling the intermediate entities in the path and modeling multiple paths. However, in their approach they have to store scores for intermediate path length for *all* entity pairs, making it prohibitive to be used in our setting where we have more than 3M entity pairs. They also model entities as just a scalar weight whereas we learn both entity and type representations. Lastly it has been shown by Neelakantan et al. (2015) that non-linear composition function out-performs linear functions (as used by them) for relation extraction tasks.

The performance of relation extraction methods have been improved by incorporating entity types for their candidate entities, both in sentence level (Roth and Yih, 2007; Singh et al., 2013) and KB relation extraction (Chang et al., 2014), and in learning entailment rules (Berant et al., 2011). Serban et al. (2016) use RNNs to generate factoid question from Freebase.

5 Results

Data and Experimental Setup

We apply our models to the dataset released by Neelakantan et al. (2015), which is a subset of Freebase enriched with information from ClueWeb. The dataset is comprised of a set of triples (e_1, r, e_2) and also the set of paths connecting the entity pair (e_1,e_2) in the knowledge graph. The triples extracted from ClueWeb consists of sentences that contain entities linked to Freebase (Orr et al., 2013). The raw text between the two entities in the sentence forms the relation

type. To limit the number of textual relations, we retain the two following words after the first entity and two words before the second entity. We also collect the entity type information from Freebase. Table 2 summarizes some important statistics. For the PathQA experiment, we use the same train/dev/test split of WordNet dataset released by Guu et al. (2015) and hence our results are directly comparable to them. The WordNet dataset has just 22 relation types and 38194 entities which is order of magnitudes less than the dataset we use for relation extraction tasks.

The dimension of the relation type representations and the RNN hidden states are $d, h = 250$ and the entity and type embeddings have $m = 50$ dimensions. The Path-RNN model has sigmoid units as their activation function. However, we found rectifier units (ReLU) to work much better (Le et al., 2015), even when compared to LSTMs (73.2 vs 72.4 in MAP). For the path-query experiment, the dimension of entity, relation embeddings and hidden units are set to 100 (as used by Guu et al. (2015)). As our evaluation metric, we use the average precision (AP) to score the ranked list of entity pairs. The MAP score is the mean AP across all query relations. AP is a strict metric since it penalizes when an incorrect entity is ranked higher above correct entities. Also MAP approximates the area under the Precision Recall curve (Manning et al., 2008). We use Adam (Kingma and Ba, 2014) for optimization for all our experiments with the default hyperparameter settings (learning rate = $1e^{-3}$, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e^{-8}$). Statistical significance for scores reported in Table 3 were done with a paired-t test.

5.1 Effect of Pooling Techniques

Section 1 of Table 3 shows the effect of the various pooling techniques presented in section 3.2. It is encouraging to see that *LogSumExp* gives the best results. This demonstrates the importance of considering information from all the paths. However, Avg. pooling performs the worst, which shows that it is also important to weigh the paths scores according to their values. Figure 3 plots the training loss w.r.t gradient update step. Due to non-zero gradient updates for all the paths, the LogSumExp pooling strategy leads to faster training vs. max pooling, which has sparse gradients. This is especially relevant during the early stages of training, where the argmax path is essentially a random guess. The scores of max and LSE pooling are significant with ($p < 0.02$).

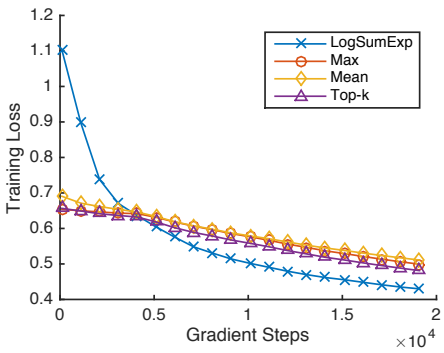

Figure 3: Comparison of the training loss w.r.t gradient update steps of various pooling methods. The loss of LogSumExp decreases the fastest among all pooling methods and hence leads to faster training.

5.2 Comparison with multi-hop models

We next compare the performance of the Single-Model with two other multi-hop models - Path-RNN and PRA(Lao et al., 2011). Both of these approaches train an individual model for each query relation. We also experiment with another extension of PRA that adds bigram features (PRA + Bigram). Additionally, we run an experiment replacing the max-pooling of Path-RNN with LogSumExp. The results are shown in the second section of Table 3. It is not surprising to see that the Single-Model, which leverages parameter sharing improves performance. It is also encouraging to see that LogSumExp makes the Path-RNN baseline stronger. The scores of Path-RNN (with LSE) and Single-Model are significant with ($p < 0.005$).

5.3 Effect of Incorporating Entities

Next, we provide quantitative results supporting our claim that modeling the entities along a KB path can improve reasoning performance. The last section of Table 3 lists the performance gain obtained by injecting information about entities. We achieve the best performance when we represent entities as a function of their annotated types in Freebase (Single-Model + Types) ($p < 0.005$). In comparison, learning separate representations of entities (Single-Model + Entities) gives slightly worse performance. This is primarily because we encounter many new entities during test time, for

Model	Performance (%MAP)	Pooling
Single-Model	68.77	Max
Single-Model	55.80	Avg.
Single-Model	68.20	Top(k)
Single-Model	**70.11**	LogSumExp
PRA	64.43	n/a
PRA + Bigram	64.93	n/a
Path-RNN	65.23	Max
Path-RNN	68.43	LogSumExp
Single-Model	**70.11**	LogSumExp
PRA + Types	64.18	n/a
Single-Model	70.11	LogSumExp
Single-Model + Entity	71.74	LogSumExp
Single-Model + Types	**73.26**	LogSumExp
Single-Model + Entity + Types	72.22	LogSumExp

Table 3: The first section shows the effectiveness of LogSumExp as the score aggregation function. The next section compares performance with existing multi-hop approaches and the last section shows the performance achieved using joint reasoning with entities and types.

which our model does not have a learned representation. However the relatively limited number of entity types helps us overcome the problem of representing unseen entities. We also extend PRA to include entity type information (PRA + Types), but this did not yield significant improvements.

5.4 Performance in Limited Data Regime

In constructing our dataset, we selected query relations with reasonable amounts of data. However, for many important applications we have very limited data. To simulate this common scenario, we create a new dataset by randomly selecting 23 out of 46 relations and removing all but 1% of the positive and negative triples previously used for training.

Effectively, the difference between Path-RNN and Single-Model is that Single-Model does multitask learning, since it shares parameters for different target relation types. Therefore, we expect it to outperform Path-RNN on this small dataset, since this multitask learning provides additional regularization. We also experiment with an extension of Single-Model where we introduce an additional task for multitask learning, where we seek to predict annotated types for entities. Here, parameters for the entity type embeddings are shared with the Single-Model. Supervision for this task is provided by the entity type annotation in the KB. We train with a Bayesian Personalized Ranking loss of Rendle et al. (2009). The results are listed in Table 4. With Single-Model there is a clear jump in performance as we expect. The additional multitask training with types gives a very incremental gain.

Model	Performance (%MAP)
Path-RNN	22.06
Single-Model	63.33
Single-Model + MTL	64.81

Table 4: Model performance when trained with a small fraction of the data.

5.5 Answering Path Queries

Guu et al. (2015) introduce a task of answering questions formulated as path traversals in a KB. Unlike binary fact prediction, to answer a path query, the model needs to find the set of correct target entities 't' that can be reached by starting from an initial entity 's' and then traversing the path 'p'. They model additive and multiplicative interactions of relations in the path. It should be noted that the compositional Trans-E and Bilinear-diag have comparable number of parameters to our model since they also represent relations as vectors, however the Bilinear model learns a dense square *matrix* for each relation and hence has a lot more number of parameters. Hence, we compare with Trans-E and Bilinear-diag models. Bilinear-diag has also been shown to outperform Bilinear models (Yang et al., 2015).

Instead of combining relations using simple additions and multiplications, we propose to combine the intermediate hidden representations h_i obtained from a RNN (via (4)) after consuming relation r_i at each step. Let $\mathbf{h}$ denote the sum of all intermediate representations h_i. The score of a triple (s, p, t) by our model is given by $x_s^\top \mathrm{diag}(\mathbf{h}) x_t$ where $\mathrm{diag}(\mathbf{h})$ represents a diagonal

Horn Clause (Body)	Without Entities	With Entities	Universal
location.contains$(x, a) \wedge$ location.contains(a, y)	0.9149	0.949	Y
(person.nationality)$^{-1}(x, a) \wedge$ place.birth(a, y)	0.7702	0.9256	N

Table 5: Body of two clauses both of which are predictive of location.contains(x, y). First fact is universally true but the truth value of the second clause depends on the value of the entities in the clause. The model without entity parameters cannot discriminate this and outputs a lower overall confidence score.

Model	MQ
Comp. Bilinear Diag	90.4
Comp. Trans-E	93.3
Our Model	**98.94**

Table 6: Performance on path queries in WordNet.

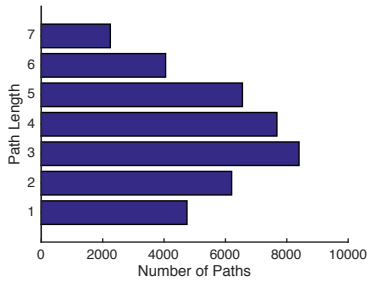

Figure 4: Length distribution of top-scoring paths

matrix with vector $\mathbf{h}$ as its diagonal elements.

We compare to the results reported by Guu et al. (2015) on the WordNet dataset. It should be noted that the dataset is fairly small with just 22 relation types and an average path length of 3.07. More importantly, there are only few *unseen paths* during test time and only *one* path between an entity pair, suggesting that this dataset is not an ideal test bed for compositional neural models. The results are shown in table 6. Mean Quantile(MQ) is the fraction of incorrect entities which have been scored lower than the correct entity. Our model achieves a 84% reduction in error when compared to their best model.

6 Qualitative Analysis

Entities as Existential Quantifiers: Table 5 shows the body of two horn clauses. Both the clauses are predictive of the fact location.contains(x, b). The first clause is *universally* true irrespective of the entities present in the chain (transitive property). However the value of the second clause is only true *conditional* on the instantiations of the entities. The score of the Path-RNN model is independent of the entity values, whereas our model outputs a different score based on the entities in the chain. We average the scores across entities, which are connected through this path and for which the relation holds in column 3 (With Entities).

For the first clause, which is independent of entities, both models predict a high score. However for the second clause, the model without entity information predicts a lower score because this path is seen in both positive and negative training examples and the model cannot condition on the entities to learn to discriminate. However our model predicts the true relations with high confidence. This is a first step towards the capturing existential quantification for logical inference in vector space.

Length of Clauses: Figure 4 shows the length distribution of top scoring paths in the test set. The distribution peaks at lengths$= \{3, 4, 5\}$, suggesting that previous approaches (Yang et al., 2015) which restrict the length to 3 might limit performance and generalizability.

Limitation: A major limitation of our model is inability to handle long textual patterns because of sparsity. Compositional approaches for modeling text (Toutanova et al., 2015; Verga et al., 2016) are a right step in this direction and we leave this as future work.

7 Conclusion

This paper introduces a single high capacity RNN model which allows chains of reasoning across multiple relation types. It leverages information from the intermediate entities present in the path between an entity pair and mitigates the problem of unseen entities by representing them as a function of their annotated types. We also demonstrate that pooling evidence across multiple paths improves both training speed and accuracy. Finally, we also address the problem of reasoning about infrequently occurring relations and show significant performance gains via multitasking.

References

Jonathan Berant, Ido Dagan, and Jacob Goldberger. 2011. Global learning of typed entailment rules. In *NAACL*.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *NIPS*.

Samuel R. Bowman, Christopher Potts, and Christopher D. Manning. 2014. Recursive neural networks for learning logical semantics. *CoRR*.

Rich Caruana. 1997. Multitask learning. *Machine Learning*.

Kai-Wei Chang, Wen tau Yih, Bishan Yang, and Christopher Meek. 2014. Typed tensor decomposition of knowledge bases for relation extraction. In *EMNLP*.

Matt Gardner, Partha Pratim Talukdar, Bryan Kisiel, and Tom M. Mitchell. 2013. Improving learning and inference in a large knowledge-base using latent syntactic cues. In *EMNLP*.

Matt Gardner, Partha Talukdar, Jayant Krishnamurthy, and Tom Mitchell. 2014. Incorporating vector space similarity in random walk inference over knowledge bases. In *EMNLP*.

Edward Grefenstette. 2013. Towards a formal distributional semantics: Simulating logical calculi with tensors. *Lexical and Computational Semantics*.

K. Guu, J. Miller, and P. Liang. 2015. Traversing knowledge graphs in vector space. In *EMNLP*.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980.

Ni Lao, Tom Mitchell, and William W. Cohen. 2011. Random walk inference and learning in a large scale knowledge base. In *EMNLP*, Stroudsburg, PA, USA.

Ni Lao, Amarnag Subramanya, Fernando Pereira, and William W. Cohen. 2012. Reading the web with learned syntactic-semantic inference rules. In *EMNLP*.

Quoc V. Le, Navdeep Jaitly, and Geoffrey E. Hinton. 2015. A simple way to initialize recurrent networks of rectified linear units. *CoRR*.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*.

Bonan Min, Ralph Grishman, Li Wan, Chang Wang, and David Gondek. 2013. Distant supervision for relation extraction with an incomplete knowledge base. In *NAACL*.

Arvind Neelakantan, Benjamin Roth, and Andrew McCallum. 2015. Compositional vector space models for knowledge base completion. In *ACL*, Beijing, China.

Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A three-way model for collective learning on multi-relational data. In *ICML*.

Dave Orr, Amar Subramanya, Evgeniy Gabrilovich, and Michael Ringgaard. 2013. 11 billion clues in 800 million documents: A web research corpus annotated with freebase concepts. `http://googleresearch.blogspot.com/2013/07/11-billion-clues-in-800-million.html`.

Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. Bpr: Bayesian personalized ranking from implicit feedback. UAI '09.

Sebastian Riedel, Limin Yao, Andrew McCallum, and Benjamin M. Marlin. 2013. Relation extraction with matrix factorization and universal schemas. In *NAACL*.

Tim Rocktäschel and Sebastian Riedel. 2016. Learning knowledge base inference with neural theorem provers. In *AKBC, NAACL*.

Dan Roth and Wen-tau Yih. 2007. Global inference for entity and relation identification via a linear programming formulation. In *In Introduction to SRL*.

Stefan Schoenmackers, Oren Etzioni, Daniel S. Weld, and Jesse Davis. 2010. Learning first-order horn clauses from web text. In *EMNLP*.

Iulian Vlad Serban, Alberto García-Durán, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 2016. Generating factoid questions with recurrent neural networks: The 30m factoid question-answer corpus. *ACL*.

Sameer Singh, Sebastian Riedel, Brian Martin, Jiaping Zheng, and Andrew McCallum. 2013. Joint inference of entities, relations, and coreference. In *AKBC, CIKM*.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. In *NIPS*.

Kristina Toutanova, Danqi Chen, Patrick Pantel, Hoifung Poon, Pallavi Choudhury, and Michael Gamon. 2015. Representing text for joint embedding of text and knowledge bases. In *EMNLP*, September.

Kristina Toutanova, Xi Victoria Lin, Scott Wen tau Yih, Hoifung Poon, and Chris Quirk. 2016. Compositional learning of embeddings for relation paths in knowledge bases and text. ACL.

Patrick Verga, David Belanger, Emma Strubell, Benjamin Roth, and Andrew McCallum. 2016. Multilingual relation extraction using compositional universal schema.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding entities and relations for learning and inference in knowledge bases. *ICLR*.

Wenyuan Zeng, Yankai Lin, Zhiyuan Liu, and Maosong Sun. 2016. Incorporating relation paths in neural relation extraction. *CoRR*.